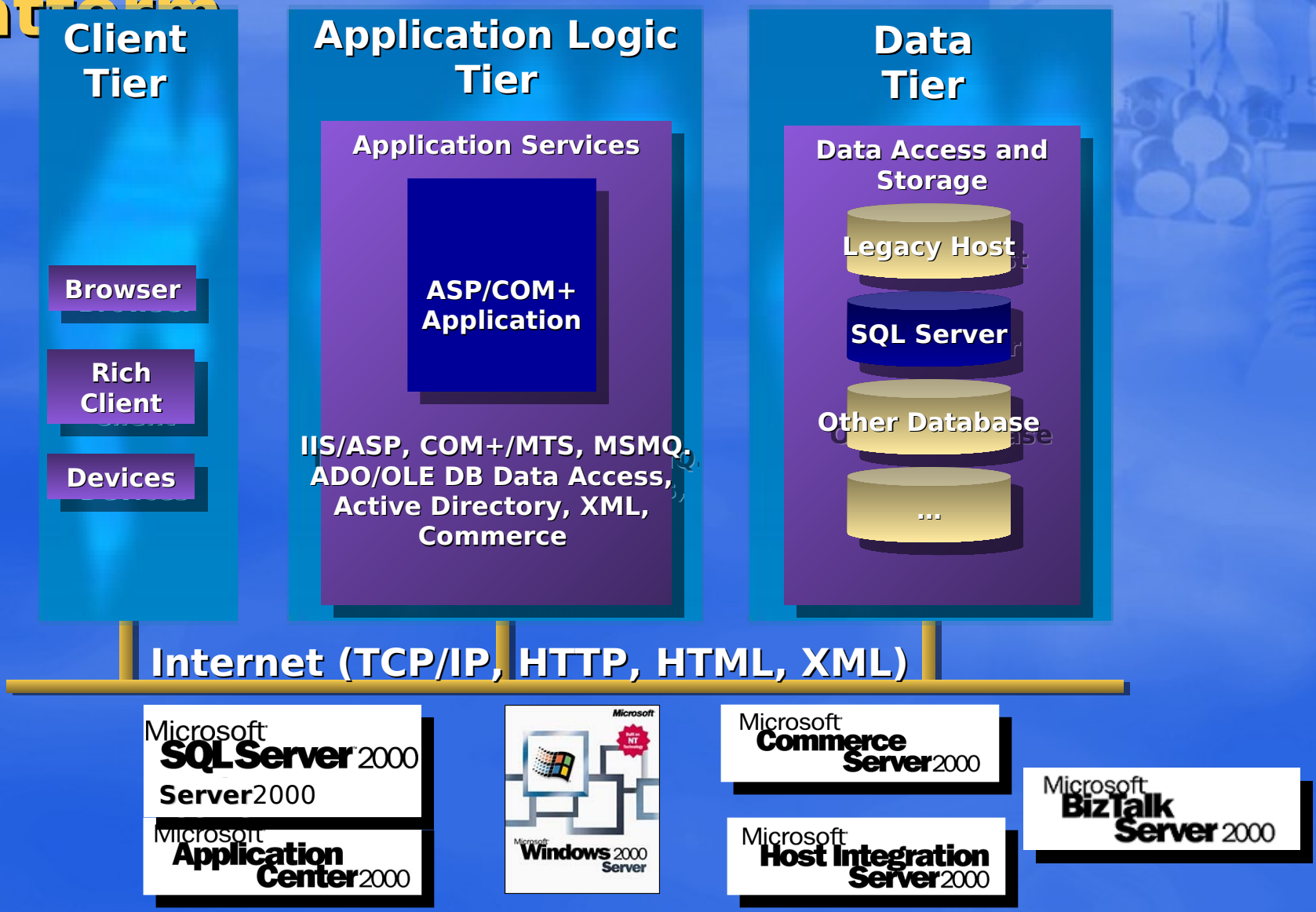
High Performance Data Access With Visual C++®

Lon Fisher
Software Design Engineer
Visual C++ Team
Microsoft Corporation


POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

Windows DNA 2000

Next Generation Web Application Platform



Agenda

- Case Study: PCWeek Benchmark
- Visual C++ Optimizations
- SQL Server™ Optimizations
- New SQL Server 2000 Optimizations

PCWeek Benchmark
Background

PCWEEK SHOOTOUT

"Application servers are the backbone of enterprise computing in the Internet economy, but it has become increasingly difficult to weigh server choices. App server vendors, whose ranks seem to grow by the day, have begun to differentiate their wares with all manner of bells and whistles. But the bottom line is performance--if the server can't keep up with demand, your e-business is in e-trouble."

Timothy Dyck for PC Week Labs

PCWeek Benchmark
Background

- PCWeek invited 8 companies to participate
  - Apple Computer (WebObjects 4.0)
  - Bluestone Software (Sapphire/Web 5.1)
  - Haht Software (Hahtsite 4.0)
  - Microsoft (Windows NT Enterprise Server 4.0)
  - Progress Software (Apptivity 3.0)
  - Sybase (Sybase EAS 3.0)
  - Sun/Netscape (NetDynamics/NAS 2.1)

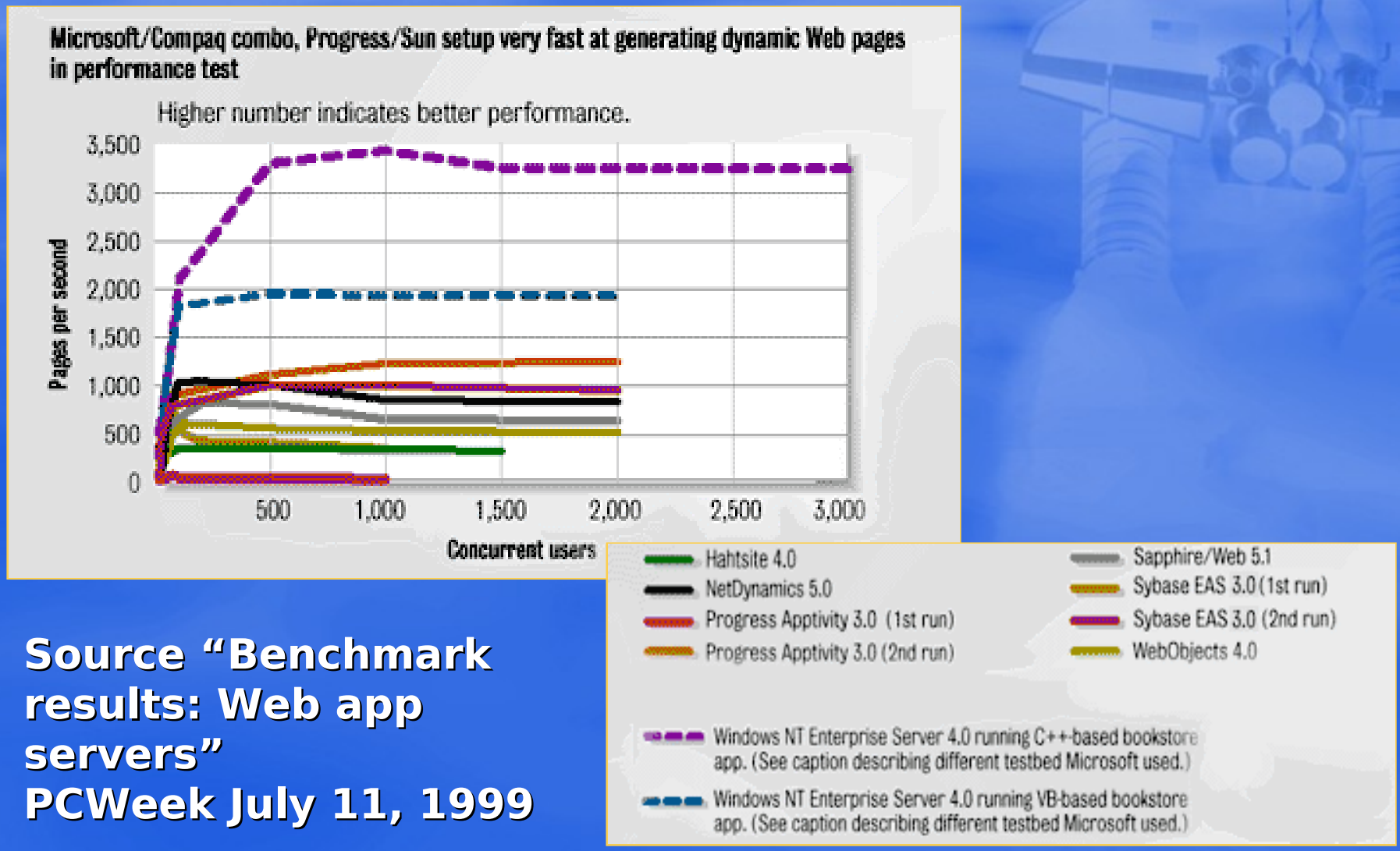
PCWeek Benchmark
The @Bench Benchmark

- **Designed to measure**
  - **Performance**
    - **Increase # users until response time exceeds 3 seconds**
    - **Measure peak pages served/sec.**
  - **Scalability**
    - **Measure capacity by adding hardware**
  - **Fault Tolerance**
    - **Pull servers, restore, and check lost transactions**
- **Must use shipping software**

PCWeek Benchmark
The @Bench Benchmark

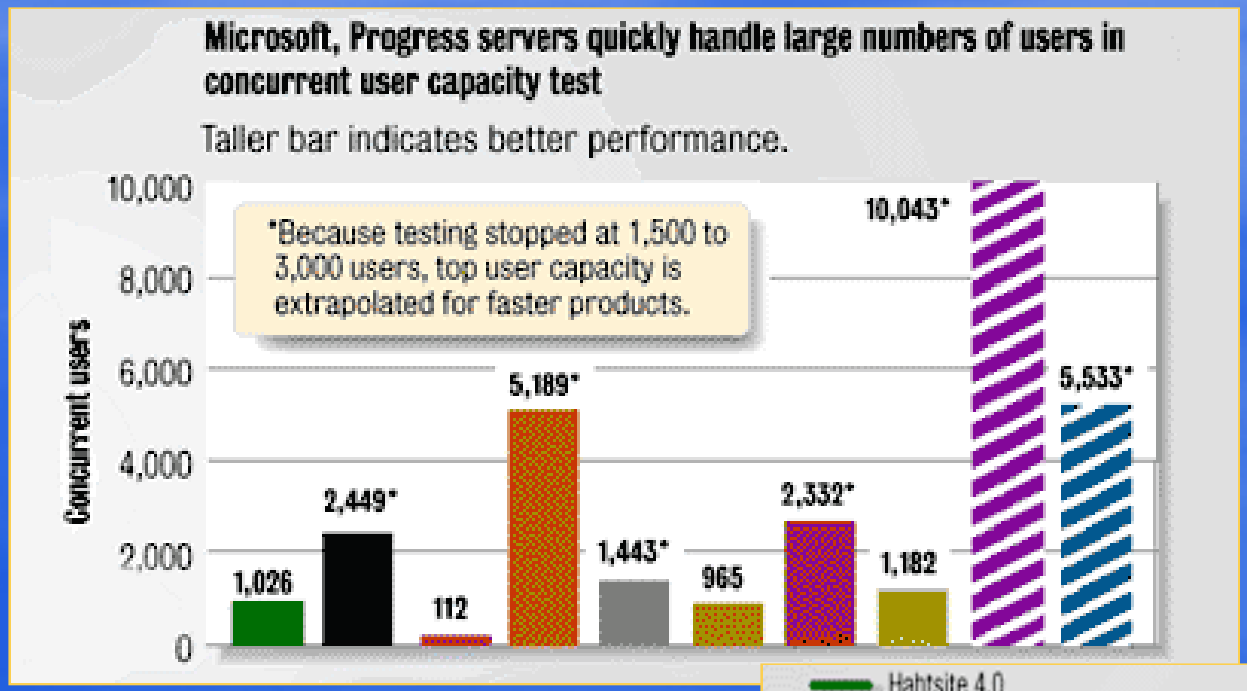
- App is an online bookstore ‘Nile.com’
- Nile.com supports
  - Browsing by subject
  - Supporting a shopping cart
  - Purchasing
  - Registering users
  - Searching by author, subject, and title
- 12.5 million row database

PCWeek Benchmark
Throughput by vendor



Source "Benchmark results: Web app servers"
PCWeek July 11, 1999

PCWeek Benchmark
User capacity by vendor



Source "Benchmark results: Web app servers"
PCWeek July 11, 1999

PCWeek Benchmark Results

PCWEEK SHOOT-OUT

"Microsoft Corp. and the Compaq Computer Corp. hardware it used came through Doculabs' tests like a bolt of lightning, blowing away everyone else's numbers--in fact, its C++-based bookstore was so fast that it was bottlenecked by the 100M-bps test network."

Timothy Dyck for PC Week Labs
July 12, 1999

PCWeek Benchmark
How did we do that?

- 50% of our results were due to good database usage
- Take full advantage of C++ and its flexibility
- Take full advantage of SQL Server

Data Access Optimization Primer

- Why is this important?
  - Can make or break your performance (50% of our performance was due to database optimizations)
  - Different methodologies have different performance implications!
- Always make best use of your database server
- Avoid making the server

Visual C++ Optimizations Data Access Methodologies

- We chose ODBC for our data access layer
  - Small, fast, works well with SQL Server
  - Arguably great performance on bulk fetching
- OLE DB would have worked well
  - We chose ODBC though because of slightly better bulk fetching semantics
- ADO works but did show a hit on Web servers
  - We noticed CPU utilization was higher with ADO

Visual C++ Optimizations

SQL Compilations

- What is a SQL Compilation?
  - SQL Server is a compiler!
  - When a statement is first called
    - Compiled to validate parameters, etc.
    - Negative performance implication
    - Utilizes more database server resources
  - After compilation, this code is cached
    - Better performance on subsequent

Visual C++ Optimizations

The wrong way to call SPs

Never call by value or call prepare!

```
// Bad because this will cause a recompilation
// unless the exact statement is cached.  If
// the value changes, so does the statement
SQLExecDirect(hStmt, “{ CALL
spfoo(‘myval’)}”, SQL_NTS);


// Also bad because it causes extra network
// without any benefit
SQLPrepare(hStmt, “{ CALL spfoo(‘myval’}”,
      SQL_NTS);
```

Visual C++ Optimizations

The eight way to call SPs

The right way to call…

```
// Bind parameters
char szParam[32];
SQLBindParameter(hStmt,…,szParam);

// Set parameters
strcpy(&szParam[0], “myval”);

// Execute the stored procedure
SQLExecDirect(hStmt, “{ CALL spTest(?) }”,
      SQL_NTS);
```

If called before, this should be in the cache

Visual C++ Optimizations

SQL Compilation demo

- Demo
  - SQL Server 7.0 ODBC application
  - Same stored procedure call
  - One bound by value the other by parameter

Visual C++ Optimizations
Bulk row fetching

- Bulk row fetching
  - Reduces network ‘wire’ hits
  - ODBC does a good job with the SET_ATTR_ROW_ARRAY_SIZE option
  - Works great with SQL Server read only, forward only cursors
    - Begins transmitting the entire result set even if rows haven’t been fetched
    - The data is already on your machine

Visual C++ Optimizations

Insertions

- We used SPs to add our data
- If you don't want to use SPs try:
  - IRowsetFastLoad (OLEDB)
  - SQLBulkOperations (ODBC)
  - Both of these offer bulk copy modes (allows you to batch insert easily)

Demo

- Fast Insertions using OLE DB
  - Use IRowsetFastLoad to ‘batch’ inserts to database

SQL Server Optimizations

Database indexes

- Database indexes
  - Think of it as a lookup table against a database
  - Usually held in memory, so they can be expensive
  - Makes it faster to retrieve rows
  - Takes longer to do an update or insert
  - Create them with the index wizard

SQL Server Optimizations

Database indexes

- **Formula for indexes**

  <ORDER BY COLUMNS>,
  <SEARCH COLUMNS>,
  <FETCH COLUMNS>

```
// For example
select job_desc, min_lvl from jobs
where job_id > ? ORDER BY max_lvl

// The index would be
max_lvl, job_id, job_desc, min_lvl
```

SQL Server Optimizations

Database indexes

- Index analysis
  - SQL Server index tuning wizard works well
  - Create a trace using SQL Server profiler
    - Sample trace will examine your indexes, just open it and you're set
  - Run the trace through the wizard

SQL Server Optimizations

Query analysis

- Query analysis
  - Understanding what a stored procedure will do is extremely important
  - SQL Server Query Analyzer is an excellent tool to help with this
  - Never do a table lookup
  - Look for expensive CPU hits
  - Be careful how many rows you need

    (<100 is good)
  - Simpler queries returning slightly

SQL Server Optimizations

Data analysis

- Data analysis
  - Always look at the data in your database
  - Great help when designing efficient stored procedures
  - For example, we had one table w/ 1.2 million records but only 600 would be valid
    - Initial SP performance was 40-60 seconds
    - Final SP performance was 150ms!

SQL Server 2000 Optimizations

- XML Support
  - Just dump it into your output stream, no need to format
- Indexed Views
  - Great when you want an already computed join on tables
- Computed Column Indexes
  - Excellent for those ID fields

Conclusions

- Little things often mean a lot!
- Visual C++ and SQL Server work great together because they are so flexible
- New SQL Server 2000 enhancements will improve indexing and Web data transfer

For Further Information

- PCWeek articles and white paper
  - Go to http://www.microsoft.com/visualc
  - Look for PCWeek article

POWER UP

Microsoft